DataDisplyControl (DDC)

Dokumentation für Vers. 3.0.1.3

Inhaltsverzeichnis

Zweck und Grundproblem ... 4
Vorteile gegenüber Standard-Controls ... 4
Einsatzgebiete ... 4
Kurzeinführung ... 5
Eigenschaften (Properties) ... 5
Methoden ... 5
Events ... 5
Arbeiten mit dem CustomDataPanel (DDC) ... 6
Hinzufügen von Elementen (AddInfoItem) ... 6
Beispiel: Hinzufügen von Elementen per Code ... 6
Aktualisieren von Elementen (UpdateInfoItem) ... 7
Beispiel: Aktualisieren von Elementen ... 7
Zusammenfassung ... 7
Aufbau und Funktionalität ... 8
Anpassung und Interaktion ... 8
Zusammenfassung der Vorteile ... 8
Aufbau des Controls ... 9
Bestandteile ... 9
a) Titelbereich ... 9
b) Rahmen ... 9
c) Hintergrund ... 9
d) Datenzeilen (MyInfoItem) ... 9
e) Items-Verwaltung ... 9
f) Interaktion ... 10
g) Automatisches Resizing ... 10
Eigenschaften des DDC ... 11
Darstellung & Layout ... 11
BackgroundColor As Color ... 11
BorderColor As Color ... 11
BorderRadius As Single ... 11
BorderWidth As Single ... 11
HoverBorderColor As Color ... 11
Inhalt & Daten ... 11
InfoItems As Collection(Of MyInfoItem) ... 11
Interaktion ... 11
ItemHoverColor As Color ... 11
Textfarben ... 12
LabelColor As Color ... 12
Title As String ... 12
TitleColor As Color ... 12
ValueColor As Color ... 12
Textschriften ... 12
TitleFont As Font ... 12
LabelFont As Font ... 12

Beispielanwendung: (Anzeige der vorhandenen Laufwerke) 13
Test-Szenario 13
Ergebnis und Interaktion 13
Beispielcode: 13
Itmes zu Laufzeit hinzugefügt: (per Laufzeit) 14
Item Nutzung als Headline: (Headline im Designer; Rest zur Laufzeit) 14
Item Nutzung als Headline2: (Event Ausschluss) 14
Das Problem mit Klick- und Doppelklick-Events 15
Verbesserte Klick- und Doppelklick-Interaktionen 15
Details der Verbesserung: 15
DDC Versionshistory 17
Kompletter Code für Version 3.0.1.2 18
Screeshoots 20

Zweck und Grundproblem

Das Data Display Control (DDC) ist eine spezialisierte Lösung für ein verbreitetes Problem in der Softwareentwicklung: die effektive Visualisierung von strukturierten Datenpunkten. Das DDC ist ein spezialisiertes WinForms-Control für .NET Framework 4.8 und .NET 8, das eine klare Visualisierung weniger, aber wichtiger Datenpunkte ermöglicht. Es dient als schlanke Alternative zu komplexen Standard-Controls wie DataGridView oder ListView.

Vorteile gegenüber Standard-Controls

Standard-Controls sind häufig für komplexe Datensätze und eine Vielzahl von Darstellungsoptionen ausgelegt, was sie oft **überdimensioniert** für einfache Aufgaben macht. Beispielsweise können DataGridViews, obwohl mächtig, unnötig kompliziert für die Anzeige von nur wenigen, aber wichtigen Datenpunkten sein. Andere Controls wie ListBoxen oder Labels bieten hingegen oft nicht die notwendigen **speziellen Funktionen** zur dynamischen Hervorhebung und Interaktion.

Das DDC füllt diese Lücke, indem es eine **klare Struktur** für die Darstellung von Daten bietet, die der Benutzer auf einen Blick erfassen kann.

- Einfachere Anzeige kleiner, relevanter Datenmengen.
- Dynamische Statusanzeige (z. B. Rot bei Fehlern).
- Interaktive Events (Click, DoubleClick, Hover).
- Automatisches Layout-Resizing je nach Anzahl der Items.

Einsatzgebiete

Das DDC eignet sich ideal für Szenarien, in denen die visuelle Lesbarkeit und ein schneller Kontext der Daten entscheidend sind. Siehe dazu die Abbildungen im Anhang.

- Dashboards (KPIs, Statusinformationen).
- Monitoring-Anwendungen (Sensorwerte, Server-Metriken).
- Gerätesteuerungen (z. B. Laufwerks- und Prozessorstatus).

Das DDC ist somit kein Ersatz für vielseitige Controls wie DataGridViews, sondern ein gezielt einsetzbares Werkzeug, das die Entwicklung von Benutzeroberflächen für spezifische, datenintensive Aufgaben effizienter und benutzerfreundlicher macht.

Kurzeinführung

Das DDC ist ein leichtgewichtiges, wiederverwendbares Control, das Datenzeilen aus Label, Wert und Einheit anzeigt. Es unterstützt Statusfarben, Auto-Resize und Benutzerinteraktion.

Eigenschaften (Properties)

Eigenschaft	Typ	Standardwert	Beschreibung
Title	String	`"Lorem Ipsum"`	Überschrift, wird oberhalb der Trennlinie dargestellt.
TitleFont	Font	Segoe UI, 11pt, Bold	Schriftart für den Titel.
ItemFont	Font	Consolas, 9pt, Regular	Schriftart für alle InfoItems (Label, Value, Unit).
BackgroundColor	Color	Transparent	Hintergrundfarbe des Panels.
BorderColor	Color	Blue	Rahmenfarbe im Normalzustand.
HoverBorderColor	Color	SteelBlue	Rahmenfarbe, wenn der Mauszeiger über dem Panel schwebt.
BorderRadius	Single	8.0	Rundung der Ecken (0 = rechteckig, höher = runder).
BorderWidth	Single	1.5	Linienstärke des Rahmens.
LabelColor	Color	(96, 96, 96)	Farbe der Beschriftungen (linke Spalte).
TitleColor	Color	(64, 64, 64)	Schriftfarbe des Titels.
ValueColor	Color	(32, 32, 32)	Standardfarbe der Werte (falls keine StatusColor gesetzt ist).
ItemHoverColor	Color	LightBlue (mit Transparenz)	Hintergrundfarbe des Items beim MouseHover.
InfoItems	Collection<MyInfoItem>	—	Liste der angezeigten InfoItems (Labels, Werte, Units). Kann im Designer oder per Code bearbeitet werden.

Methoden

Methode	Parameter	Beschreibung
AddInfoItem(label, value, unit)	String label, String value, String unit	Fügt ein neues Item mit Standardfarbe (Grau) hinzu.
AddInfoItem(label, value, color, unit)	String label, String value, Color statusColor, String unit	Fügt ein neues Item mit einer Statusfarbe hinzu.
ClearInfoItems()	—	Entfernt alle Items aus dem Panel.
UpdateInfoItem(index, newValue, [newUnit], [newStatusColor])	Integer index, String newValue, optional String newUnit, optional Color	Aktualisiert ein Item anhand seiner Position in der Liste.
UpdateInfoItem(label, newValue, [newStatusColor])	String label, String newValue, optional Color	Aktualisiert ein Item anhand seines Labels.

Events

Event	Argumente	Beschreibung
ItemClick	InfoItemEventArgs (Item, Index)	Wird ausgelöst, wenn ein Item im Panel angeklickt wird.
ItemDoubleClick	InfoItemEventArgs (Item, Index)	Wird ausgelöst, wenn ein Item im Panel doppelt angeklickt wird.
PanelClick	sender As Object, e As MouseEventArgs	Wird ausgelöst, wenn das Panel angeklickt wird.
PanelDoubleClick	sender As Object, e As EventArgs	Wird ausgelöst, wenn das Panel doppelt angeklickt wird.

Arbeiten mit dem CustomDataPanel (DDC)

Das DDC kann nicht nur im Designer mit festen Werten befüllt werden, sondern auch **dynamisch per Code**. Dies ist besonders nützlich, wenn Daten aus Hardware-Sensoren, Services oder externen Quellen in Echtzeit angezeigt werden sollen.

Im Folgenden wird Schritt für Schritt erklärt, wie Sie das Panel zur Laufzeit **füllen** und **aktualisieren** können.

Hinzufügen von Elementen (AddInfoItem)

Um neue Informationszeilen (sogenannte *InfoItems*) in das Panel aufzunehmen, verwenden Sie die Methode `AddInfoItem`.
Diese Methode ist überladen und erlaubt unterschiedliche Szenarien:

- **Nur Text und Einheit** → Standardfarbe (Grau) wird verwendet.
- **Text + Einheit + Statusfarbe** → eine Signalfarbe (z. B. Grün/Rot) wird direkt für den Wert gesetzt.

Beispiel: Hinzufügen von Elementen per Code

```
    Public Sub AddItemsExample()
' Verwenden Sie ein bereits im Form Designer vorhandene DDC.

' Löschen  alle bestehenden Elemente, um sicherzustellen, dass das Panel leer ist.
        myDataPanel.ClearInfoItems()

' Fügen Sie ein Element ohne Statusfarbe hinzu. Gray wäre dann Standard!
        myDataPanel.AddInfoItem("Temperatur", "25.4", "°C")

' Fügen Sie ein Element mit einer Statusfarbe (grün) hinzu, um einen "guten" Status zu signalisieren.
        myDataPanel.AddInfoItem("Luftfeuchte", "45", Color.Green, "%")

' Fügen Sie ein weiteres Element mit einer Statusfarbe (rot) hinzu, um einen kritischen Status anzuzeigen.
        myDataPanel.AddInfoItem("System-Last", "98", Color.Red, "%")

' Das Panel passt seine Größe automatisch an den Inhalt an.
    End Sub
```

In diesem Beispiel werden drei Messwerte dargestellt: Temperatur, Luftfeuchtigkeit und Systemlast. Die Temperatur wird neutral angezeigt, während die Luftfeuchtigkeit und die Auslastung mit Farben hervorgehoben werden.

Aktualisieren von Elementen (UpdateInfoItem)

Einmal hinzugefügte Elemente können Sie jederzeit **aktualisieren**, ohne die gesamte Liste neu aufzubauen. Dies geschieht mit der Methode `UpdateInfoItem`, die wahlweise über den **Index** oder über das **Label** des Items arbeitet.

Beispiel: Aktualisieren von Elementen

```
Public Sub UpdateItemsExample()
    ' Nehmen wir an, das Panel wurde bereits mit den obigen Elementen befüllt.

    ' Aktualisieren Sie den Wert der "Temperatur" mit dem Label.
    myDataPanel.UpdateInfoItem("Temperatur", "25.9")

    ' Aktualisieren Sie den Wert und die Statusfarbe der "System-Last" mit dem Label.
    ' Ändern Sie die Farbe auf Gelb, wenn die Last sinkt.
    myDataPanel.UpdateInfoItem("System-Last", "75", Color.Yellow)

    ' Aktualisieren Sie ein Element über seinen Index (das zweite Element hat Index 1).
    ' Ändern Sie die "Luftfeuchte" und die Einheit.
    myDataPanel.UpdateInfoItem(1, "48", "g/m³")

     ' Beachten Sie, dass das `CustomDataPanel` automatisch `Invalidate()` aufruft, um die Neuzeichnung zu erzwingen.
End Sub
```

Dieses Beispiel zeigt, wie Sie die Werte und Farben der bereits hinzugefügten Elemente ändern können. Die Methode `UpdateInfoItem` ermöglicht es Ihnen, auf spezifische Elemente zuzugreifen, entweder durch ihren eindeutigen Bezeichner (Label) oder ihre Position (Index), was eine effiziente Datenaktualisierung in Echtzeit ermöglicht.
Dieses Beispiel zeigt die drei häufigsten Anwendungsfälle:

- Aktualisieren über den **Bezeichner (Label)**
- Aktualisieren über den **Index** (Position in der Liste)
- Gleichzeitig den Wert und die **Farbe** anpassen

Zusammenfassung

Mit den Methoden `AddInfoItem`, `ClearInfoItems` und `UpdateInfoItem` können Sie das Panel flexibel einsetzen:

- Es ist möglich, **statisch definierte Daten** (z. B. Geräteparameter) anzuzeigen.
- Ebenso können **dynamische Statuswerte** (z. B. Sensoren, Laufwerksstatus, Monitoring-Daten) in Echtzeit aktualisiert werden.
- Durch die integrierte Auto-Resize-Funktion passt sich das Panel automatisch an die Menge der Einträge an.

Das **DDC** eignet sich daher sowohl für **feste Anzeigeelemente** (z. B. CPU/RAM/Netzwerk-Infos) als auch für **laufend wechselnde Werte** (z. B. Fortschritte, Statusanzeigen). Automatisches Resize sorgt dafür, dass das Panel immer passend groß ist. Mit den Events ItemClick und ItemDoubleClick lassen sich Interaktionen komfortabel umsetzen.

Aufbau und Funktionalität

Das DDC nutzt eine objektorientierte Struktur, um Daten zu verwalten:

- **MyInfoItem-Objekte**: Jedes angezeigte Datenelement ist ein **`MyInfoItem`-Objekt**. Es enthält die drei Hauptbestandteile, die für die Darstellung einer Informationseinheit erforderlich sind: ein **Label** (die Beschriftung), einen **Value** (den eigentlichen Wert) und eine **Unit** (die Einheit). Ein Beispiel dafür ist „Temperatur: 25,5 °C“.
- **StatusColor-Eigenschaft**: Eine der wichtigsten Funktionen ist die **`StatusColor`-Eigenschaft**. Sie erlaubt die dynamische Änderung der Farbe von Wert und Einheit in jeder Zeile, basierend auf dem Status der Daten. Das ermöglicht es dem Benutzer, sofort zu erkennen, ob ein Wert „gut“ (z.B. grün), „kritisch“ (z.B. rot) oder eine „Warnung“ (z.B. gelb) darstellt, ohne die Zahlen interpretieren zu müssen.
- **Datenverwaltung**: Die Daten werden in einer speziellen Liste namens **`InfoItems`-Sammlung** verwaltet. Diese Liste ist als **`BindingList`** implementiert, was bedeutet, dass Änderungen an den Daten automatisch auf der Benutzeroberfläche aktualisiert werden. Sie können Elemente zur Entwurfszeit im Editor oder zur Laufzeit per Code hinzufügen, aktualisieren oder löschen.
- **Automatisches Layout**: Das Control verfügt über eine **`ResizeToContent`-Funktion**, die die Höhe des Steuerelements automatisch an die Anzahl der enthaltenen Elemente anpasst. Das erspart dem Entwickler das manuelle Schreiben von Layout-Code und sorgt für ein sauberes Erscheinungsbild, unabhängig von der Datenmenge.

Anpassung und Interaktion

- **Umfassende visuelle Anpassung**: Das DDC ist hochgradig anpassbar, um eine nahtlose Integration in das bestehende UI-Design zu gewährleisten. Man kann Titel und Rahmen anpassen, einschließlich Text, Farben, Rahmenstärke, Eckenradius und einer **`ItemHoverColor`**. Auch die Hintergrundfarbe ist von transparent bis vollflächig veränderbar.
- **Interaktive Elemente**: Das Control reagiert auf Benutzerinteraktionen. Es löst Events wie **`ItemClick`** und **`ItemDoubleClick`** aus, wenn auf eine Zeile geklickt wird. So kann auf die Aktionen des Nutzers reagiert werden, wie z.B. das Öffnen eines Verzeichnisses bei einem Doppelklick.

Zusammenfassung der Vorteile

Das DDC ist keine generische Datenansicht, sondern ein fokussiertes Tool, das den visuellen Lärm reduziert und die wichtigsten Informationen hervorhebt.

- Fokussiertes Tool statt generisches Grid.
- Visuelle Klarheit durch Statusfarben.
- Nahtlose Integration ins UI durch anpassbares Design.
- Einfache Echtzeit-Updates der Daten.
- Interaktive Benutzerführung durch Events.

Aufbau des Controls

Ziel des DDC ist es, eine benutzerdefinierte WinForms-Benutzeroberfläche (.NET Framework 4.8 / .NET 8) bereitzustellen. Mit diesem User-Control können Sie strukturierte Daten übersichtlich in Form von Paaren aus Beschriftung, Wert und Einheit anzeigen. Es wurde speziell für die Verwendung in Dashboards, Monitor-Anzeigen, System- und Statusinformationen entwickelt, um eine klare und effiziente Visualisierung von Daten zu ermöglichen.

Bestandteile

a) Titelbereich

- Eigenschaft `Title` (String) → Überschrift des Panels
- Eigenschaft `TitleColor` (Color) → Farbe der Überschrift
- Schriftart: `"Segoe UI", 11, Bold`
- `Durch eine Linie vom Datembereich abgegrenzt`

b) Rahmen

- Eigenschaft `BorderColor` (Color) → normale Rahmenfarbe
- Eigenschaft `HoverBorderColor` (Color) → Rahmenfarbe bei Hover
- Eigenschaft `BorderRadius` (Single) → Eckenradius
- Eigenschaft `BorderWidth` (Single) → Rahmenstärke

c) Hintergrund

- Eigenschaft `BackgroundColor` (Color) → Panel-Hintergrund
- Optional Transparenz oder Vollfarbe

d) Datenzeilen (`MyInfoItem`)

Jede Zeile besteht aus:

- **Label** (z. B. „Laufwerk C:“)
- **Value** (z. B. „120 / 256“)
- **Unit** (z. B. „GB“)
- **StatusColor** (Color) → beeinflusst Value- und Unit-Farbe

Durch `ExpandableObjectConverter` werden die Items im Designer-PropertyGrid editierbar.

e) Items-Verwaltung

- Property `InfoItems As BindingList(Of MyInfoItem)` → Sammlung aller Items
- Methoden:
    - `AddInfoItem(label As String, value As String, unit As String)`
    - `AddInfoItem(label As String, value As String, statusColor As Color, unit As String)`
    - `UpdateInfoItem(index As Integer, newValue As String, newUnit As String, newStatusColor As Color)`
    - `ClearInfoItems()`

f) Interaktion

- Events:
    - `ItemClick(sender, e As InfoItemEventArgs)`
    - `ItemDoubleClick(sender, e As InfoItemEventArgs)`
- Eigenschaft `ItemHoverColor As Color` → Hintergrundfarbe beim Überfahren eines Items mit der Maus

g) Automatisches Resizing

- Panel passt seine Höhe automatisch an die Anzahl der Items an.
- Methode `ResizeToContent()` wird bei Änderungen getriggert.

Eigenschaften des DDC

Annahme: Wir haben das Nuget-Paket installiert und das Control mittels Designer aus der Toolbox auf das Form gezogen. Nun können wir im Eigenschaften Fenster folgende Werte setzen:

Darstellung & Layout

BackgroundColor As Color

- **Bedeutung:** Hintergrundfarbe des Panels (innen).
- **Default:** `Transparent`
- **Beispiel:** Hellgrau (`Color.LightGray`) für Dashboard-Optik.

BorderColor As Color

- **Bedeutung:** Rahmenfarbe des Controls im Normalzustand.
- **Default:** `Color.Blue`
- **Beispiel:** `Color.DarkGray` für dezente Umrandung.

BorderRadius As Single

- **Bedeutung:** Rundung der Ecken.
- **Default:** `8.0`
- **Wertbereich:** `0` = eckig, größer = stärker abgerundet.

BorderWidth As Single

- **Bedeutung:** Stärke (Dicke) der Rahmenlinie.
- **Default:** `1.5`

HoverBorderColor As Color

- **Bedeutung:** Rahmenfarbe, wenn die Maus über dem Panel schwebt.
- **Default:** `Color.SteelBlue`
- **Nutzen:** Hebt das Panel bei Interaktion hervor.

Inhalt & Daten

InfoItems As Collection(Of MyInfoItem)

- **Bedeutung:** Sammlung von Datenzeilen, die im Panel angezeigt werden.
- **Typ:** `MyInfoItem` (besteht aus `Label`, `Value`, `Unit`, `StatusColor`).
- **Editor:** Collection-Editor im Designer → mehrere Items hinzufügen/editieren.
- **Beispiel:**
    - `Label = "CPU"`
    - `Value = "35"`
    - `Unit = "%"`
    - `StatusColor = Color.Green`

Interaktion

ItemHoverColor As Color

- **Bedeutung:** Hintergrundfarbe, wenn die Maus über einem einzelnen Item steht.
- **Default:** `Color.FromArgb(40, Color.LightBlue)`
- **Nutzen:** Gibt visuelles Feedback für Klick-/Doppelklick-Events.

Textfarben

`LabelColor As Color`

- **Bedeutung:** Textfarbe für die Labels (linke Spalte).
- **Default:** `Color.FromArgb(96, 96, 96)`

`Title As String`

- **Bedeutung:** Titel (Überschrift) des Panels.
- **Default:** `"Lorem Ipsum Dolores"`

`TitleColor As Color`

- **Bedeutung:** Textfarbe des Titels.
- **Default:** `Color.FromArgb(64, 64, 64)`

`ValueColor As Color`

- **Bedeutung:** Standardfarbe für Werte (rechte Spalte).
- **Default:** `Color.FromArgb(32, 32, 32)`
- **Hinweis:** Kann durch `StatusColor` pro Item überschrieben werden.

Textschriften

`TitleFont As Font`

- **Bedeutung:** Titel (Überschrift) des Panels.
- **Default:** `Segoe UI Semibold;11pt`

`LabelFont As Font`

- **Bedeutung:** Label-Wert-Einheit der Items
- **Default:** `Consolas 9pt`

Beispielanwendung: (Anzeige der vorhandenen Laufwerke)

Test-Szenario

Ein praktisches **Anwendungsbeispiel** für das DDC ist die Darstellung von **Laufwerksinformationen**. Dabei werden der freie und der gesamte Speicherplatz jedes Laufwerks auf dem System visualisiert. Mithilfe der **.NET `DriveInfo`-Klasse** liest ein Beispielcode alle verfügbaren Laufwerke aus. Für jedes Laufwerk wird ein `MyInfoItem`-Objekt erstellt, das den Buchstaben, den belegten und gesamten Speicherplatz sowie die Einheit (Gigabyte) anzeigt.

- **Fehlerbehandlung**: Sollte ein Laufwerk nicht bereit sein (z.B. ein leeres CD-Laufwerk), fängt der Code dies ab und zeigt anstelle der Speicherdaten die Meldung „**Laufwerk nicht bereit**“ an. Die `StatusColor` wird dann auf Rot gesetzt, um den Fehler sofort zu visualisieren.
- **Doppelklick-Aktion**: Ein Doppelklick auf ein Laufwerks-Item öffnet den Windows Explorer für das entsprechende Laufwerk, was eine direkte Benutzerinteraktion ermöglicht.

Ergebnis und Interaktion

Die Informationen werden im Panel in einer Liste angezeigt. Jedes **Laufwerk** wird als eigenes **Element (Item)** dargestellt, das folgende Funktionen bietet:

- **Visuelles Feedback beim Überfahren:** Wenn Sie mit dem Mauszeiger über ein Laufwerk-Item fahren, ändert sich die Hintergrundfarbe leicht, um hervorzuheben, welches Element gerade ausgewählt ist.
- **Aktion per Klick:** Ein **Doppelklick** auf ein Laufwerkselement öffnet den Windows-Explorer und navigiert direkt zum entsprechenden Laufwerk. Diese Aktion kann alternativ auch mit einem **Einfachklick** ausgeführt werden, je nach Konfiguration des Code.

Beispielcode:

```
Imports DataDisplayControl
Public Class FrmMain

    Public Sub New()
        InitializeComponent()
    End Sub

    Private Sub FrmMain_Load(sender As Object, e As EventArgs) Handles MyBase.Load

        ' Testdaten (alle Laufwerke)
        For Each d As DriveInfo In DriveInfo.GetDrives()
            Try
                Dim freeGB As Double = Math.Round(d.AvailableFreeSpace / (1024 ^ 3), 1)
                Dim totalGB As Double = Math.Round(d.TotalSize / (1024 ^ 3), 1)
                CustomDataPanel1.AddInfoItem(
                    d.Name,
                    freeGB.ToString() & " / " & totalGB.ToString(), Color.AliceBlue,"GB"
                )
            Catch ex As Exception
                CustomDataPanel1.AddInfoItem(d.Name, "nicht verfügbar", Color.Red, "")
            End Try
        Next

        Me.Controls.Add(CustomDataPanel1)
    End Sub

End Class
```

Itmes zu Laufzeit hinzugefügt: (per Laufzeit)

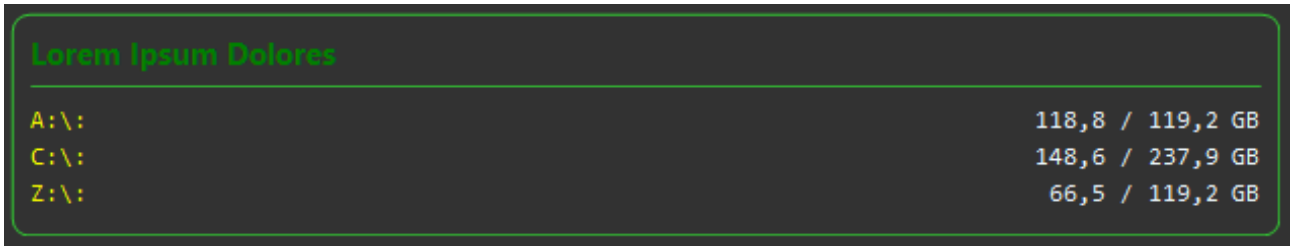

Abbildung 1: Das 'DDC' zur Laufzeit

Item Nutzung als Headline: (Headline im Designer; Rest zur Laufzeit)

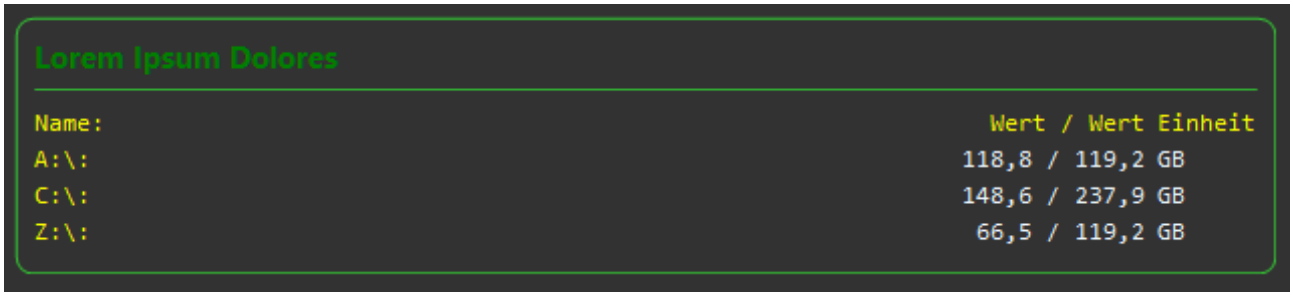

Abbildung 2: DDC mit Item als Headline

Item Nutzung als Headline2: (Event Ausschluss)

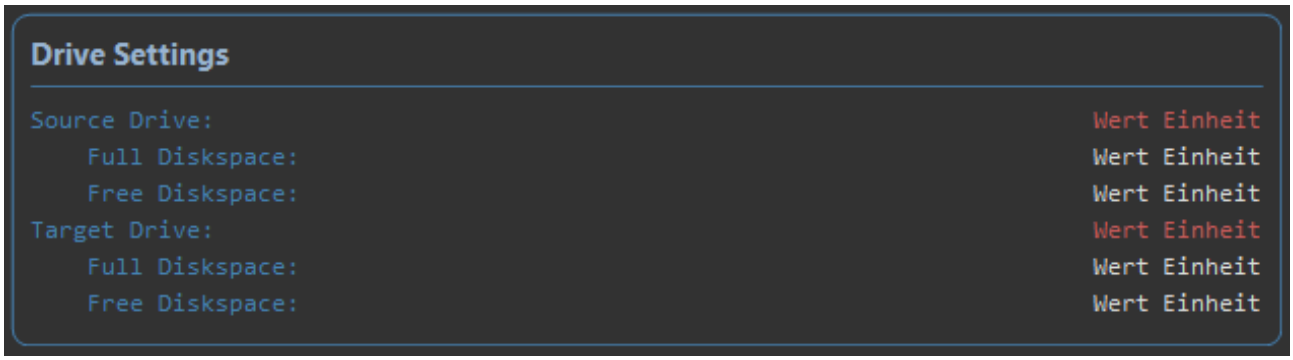

Abbildung 3: DDC mit Item als 2te Headline und Eventausschluss

Das Problem mit Klick- und Doppelklick-Events

Wenn Sie einen Doppelklick auf ein Element ausführen, werden in der Regel nacheinander die folgenden Ereignisse ausgelöst:

1. Ein **Klick-Event** (auch bekannt als `MouseClick` oder `ItemClick`)
2. Ein weiteres **Klick-Event**
3. Ein **Doppelklick-Event** (auch bekannt als `DblClick` oder `ItemDoubleClick`)

Bei einem einfachen Klick wird die Aktion sofort ausgeführt, noch bevor ein möglicher Doppelklick erkannt wird. Klickt der Nutzer doppelt, erscheint dadurch zuerst kurz die SingleClick-Aktion (z. B. eine Benachrichtigung), bevor die eigentliche Doppelklick-Aktion startet. Das wirkt verwirrend und unterbricht den Arbeitsfluss.

Die gleichzeitige Verwendung von Klick- (`ItemClick`) und Doppelklick-Events (`ItemDoubleClick`) auf einem Steuerelement kann zu ungewolltem Verhalten führen, da ein Doppelklick immer aus zwei schnellen Klicks besteht und der Klick-Event somit zuerst ausgelöst wird. Um dies zu vermeiden, ist es am effektivsten, die Funktionalitäten für jeden Event-Typ klar voneinander zu trennen.

Verbesserte Klick- und Doppelklick-Interaktionen

In dieser **Version 3.0.1.1** wurde das Eventhandling für **ItemClick** und **ItemDoubleClick** in `CustomDataPanel` wesentlich verbessert, um eine zuverlässige Unterscheidung zwischen einfachem Klick und Doppelklick zu ermöglichen. Vorher wurden Doppelklicks oft als zwei aufeinanderfolgende SingleClicks interpretiert, was zu unerwarteten Reaktionen in der Benutzeroberfläche führte.

Details der Verbesserung:

1. **Einführung eines internen Timers:**
    - Ein `System.Windows.Forms.Timer` wird intern genutzt, um zwischen Single- und DoubleClick zu unterscheiden.
    - Beim ersten Klick wird der Timer gestartet. Vergeht die festgelegte DoubleClick-Intervalle (System.Windows.Forms.SystemInformation.DoubleClickTime) ohne zweiten Klick, wird das Event als SingleClick ausgelöst.
    - Kommt ein zweiter Klick innerhalb des Intervalls, wird der Timer gestoppt und stattdessen das DoubleClick-Event ausgelöst.
2. **MouseEventArgs in ItemClick-Events:**
    - `InfoItemEventArgs` wurde erweitert, sodass neben `Item` und `Index` nun auch `MouseButtons` und `Clicks` verfügbar sind.
    - Dadurch können Anwendungen direkt auf **rechte oder linke Maustaste** reagieren oder die Klickanzahl auswerten, ohne zusätzliche Logik im Formular zu implementieren.
3. **Vermeidung von Doppel-Fehlern:**
    - Früher führte ein Doppelklick immer zuerst zu einem SingleClick. Nun wird eindeutig unterschieden.
    - Entwickler können im Event-Handler gezielt **SingleClick** oder **DoubleClick** behandeln, ohne zusätzliche Timer- oder Verzögerungslogik im Formular selbst schreiben zu müssen.
4. **Kompatibilität:**
    - Bestehende Anwendungen, die SingleClick-Events verwendeten, bleiben unverändert funktionsfähig.
    - Neue Features wie die Auswertung der Maustaste (`MouseButtons`) oder Klickanzahl (`Clicks`) können optional genutzt werden.

Beispiel für die Nutzung im Formular:

```
    ' SingleClick: einfache Aktion, z. B. MessageBox oder Log
       Private Sub myDDC1_ItemClick(sender As Object, e As InfoItemEventArgs) Handles
myDDC1.ItemClick

MessageBox.Show($"SingleClick auf Item '{e.Item.Label}' (Index {e.Index})",
                        "SingleClick", MessageBoxButtons.OK, MessageBoxIcon.Information)
    End Sub


    ' DoubleClick: z. B. Explorer öffnen
     Private Sub myDDC1_ItemDoubleClick(sender As Object, e As InfoItemEventArgs) Handles
myDDC1.ItemDoubleClick

        Try
            Dim path As String = e.Item.Label
            ' Prüfen, ob Pfad existiert
            If Directory.Exists(path) OrElse File.Exists(path) Then
                Process.Start("explorer.exe", path)
            Else
                MessageBox.Show($"Pfad nicht gefunden: {path}",
                                "Fehler", MessageBoxButtons.OK, MessageBoxIcon.Warning)
            End If
        Catch ex As Exception
            MessageBox.Show("Fehler beim Öffnen: " & ex.Message,
                            "Fehler", MessageBoxButtons.OK, MessageBoxIcon.Error)
       End Try
    End Sub
```

Vorteile für Entwickler:

- Keine falschen SingleClick-Auslösungen mehr bei DoubleClicks.
- Vollständige Kontrolle über Klicktyp, Maustaste und Zielobjekt.
- Sauberer und konsistenter Eventfluss, direkt im `CustomDataPanel` implementiert.

DDC Versionshistory

- 1.0.0.0 – Erste Veröffentlichung mit statischem Layout.
- 2.0.0.0 – Erweiterung um Rahmenoptionen (Farbe, Hover, Radius).
- 2.0.1.0 – Einführung von Klick- und Hover-Events.
- 3.0.0.0 – Aktuelle Version mit Auto-Resize, StatusColor und erweiterten Interaktionen.
- 3.0.1.1 – Verbesserte Klick- und Doppelklick-Logik (neues Codbeispiel im Anhang)
- 3.0.1.3 – Verbesserte Übersicht im Designer-Eingenschafts-Fenster vom VS
- 3.0.1.4 – Kleiner interne Umbenennungen und Optimierungen

Kompletter Code für Version 3.0.1.2

(neue Klick & DblKlick Events für die Item)

```
Imports System.IO
Imports DataDisplayControl

Public Class FrmMain

    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        myDDC1.Title = "DDC-Example Version: 3.0.1.3"
    End Sub

    ' ---------------- Demo Buttons ----------------
    Private Sub BtnAddItems_Click(sender As Object, e As EventArgs) Handles BtnAddItems.Click
        AddItemsExample()
    End Sub

      Private Sub BtnUpdateItems_Click(sender As Object, e As EventArgs) Handles BtnUpdateI-
tems.Click
        UpdateItemsExample()
    End Sub

    Private Sub BtnGetDrives_Click(sender As Object, e As EventArgs) Handles BtnGetDrives.Click
        GetInstalledDrives()
    End Sub


    ' ---------------- Demo Methoden ----------------
    Public Sub AddItemsExample()
        BtnUpdateItems.Enabled = True
        myDDC1.Title = "DDC-Example-ADD"
        myDDC1.ClearInfoItems()

        ' Beispiel-Items
        myDDC1.AddInfoItem("Temperatur", "25.4", "°C")
        myDDC1.AddInfoItem("Luftfeuchte", "45", Color.Green, "%")
        myDDC1.AddInfoItem("System-Last", "98", Color.Red, "%")
    End Sub

    Public Sub UpdateItemsExample()
        myDDC1.Title = "DDC-Example-Update"
        myDDC1.UpdateInfoItem("Temperatur", "25.9")
        myDDC1.UpdateInfoItem("System-Last", "75", Color.Blue)
        myDDC1.UpdateInfoItem(1, "48", "g/m³")
    End Sub



    ' Farben dynamisch setzen – DEMO, Schwellenwerte anpassbar
    Private Function GetDriveColor(freeGB As Double, totalGB As Double) As Color
        If totalGB <= 0 Then Return Color.DarkGray
        Dim usagePercent As Double = (totalGB - freeGB) / totalGB * 100

        If usagePercent < 40 Then
            Return Color.Green
        ElseIf usagePercent < 50 Then
            Return Color.Orange
        Else
            Return Color.Red
        End If
    End Function
```

```
    ' ---------------- Laufwerke anzeigen ----------------
    Private Sub GetInstalledDrives()
        BtnUpdateItems.Enabled = False
        myDDC1.Title = "DDC-Get-Drives"
        myDDC1.ClearInfoItems()

        ' Headline
        myDDC1.AddInfoItem("Name / Beschreibung", "Wert", Color.Black, "Einheit")

        ' Laufwerke
        For Each d As DriveInfo In DriveInfo.GetDrives()
            Try
                Dim freeGB As Double = Math.Round(d.AvailableFreeSpace / (1024 ^ 3), 1)
                Dim totalGB As Double = Math.Round(d.TotalSize / (1024 ^ 3), 1)
                Dim statusColor As Color = GetDriveColor(freeGB, totalGB)

                myDDC1.AddInfoItem(d.Name, $"{freeGB} / {totalGB}", statusColor, "GB")
            Catch ex As Exception
                myDDC1.AddInfoItem(d.Name, "nicht verfügbar", Color.Red, "")
            End Try
        Next

        ' Sicherstellen, dass Control gezeichnet wird
        Me.Controls.Add(myDDC1)
    End Sub


    ' ---------------- Klick-Handler ----------------
    ' SingleClick – Headline (Index 0) ignorieren
     Private Sub myDDC1_ItemClick(sender As Object, e As InfoItemEventArgs) Handles myDDC1.Item-
Click
        If e.Index = 0 Then Return
        MessageBox.Show($"SingleClick auf Item '{e.Item.Label}' (Index {e.Index})",
                        "SingleClick",
                        MessageBoxButtons.OK,
                        MessageBoxIcon.Information)
    End Sub



    ' DoubleClick – Explorer öffnen, Headline ignorieren
       Private Sub myDDC1_ItemDoubleClick(sender As Object, e As InfoItemEventArgs) Handles
myDDC1.ItemDoubleClick
        If e.Index = 0 Then Return
        Try
            Dim path As String = e.Item.Label
            If Directory.Exists(path) OrElse File.Exists(path) Then
                Process.Start("explorer.exe", path)
            Else
                  MessageBox.Show($"Pfad nicht gefunden: {path}", "Fehler", MessageBoxButtons.OK,
MessageBoxIcon.Warning)
            End If
        Catch ex As Exception
             MessageBox.Show("Fehler beim Öffnen: " & ex.Message, "Fehler", MessageBoxButtons.OK,
MessageBoxIcon.Error)
        End Try
    End Sub

End Class
```

Screeshoots

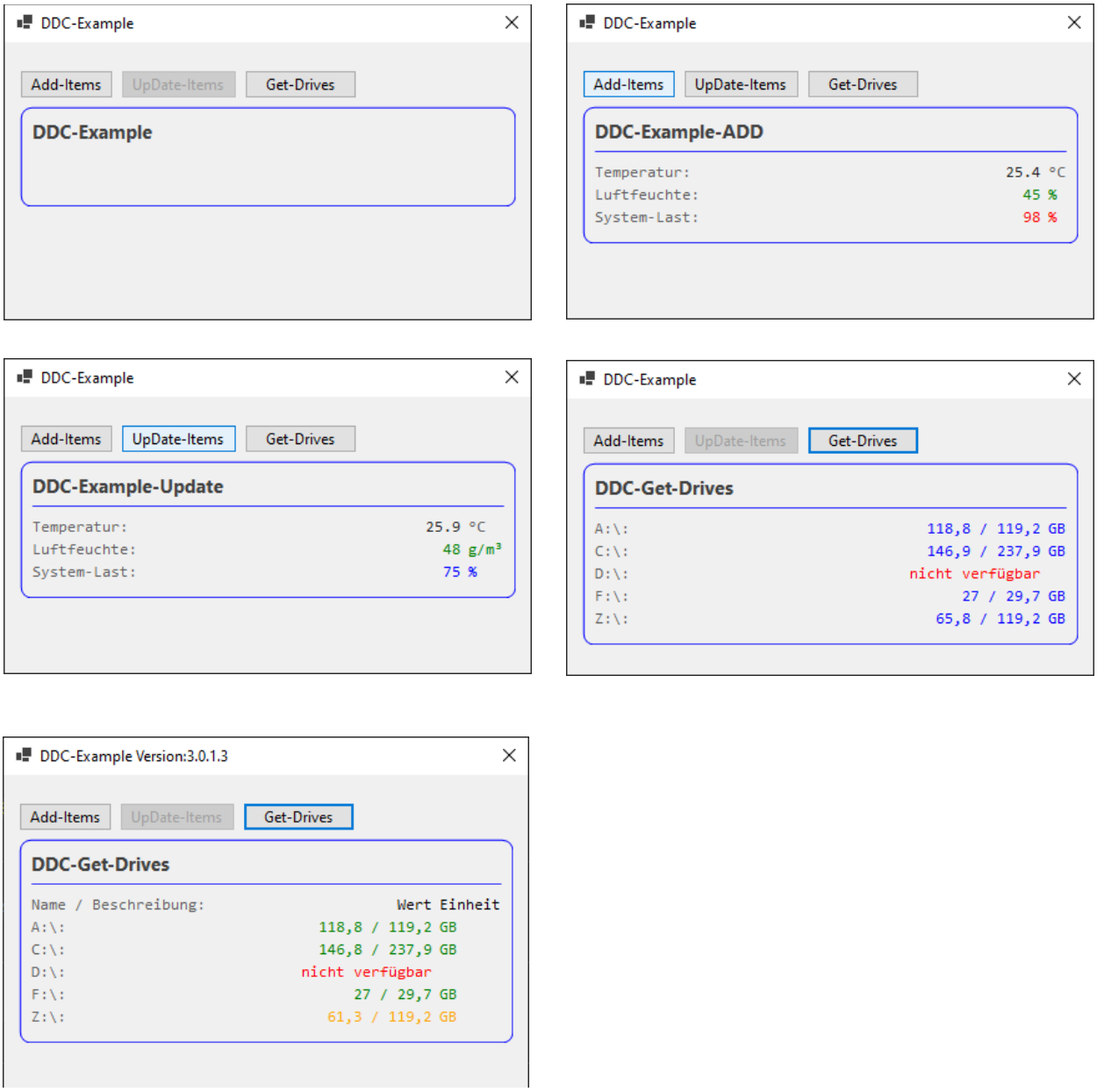

Abbildung 4: Hier mit dynamischer Farbzuweisung